When Heuristics Backfire

Description: Which Linux distro is selling itself to private equity capital, and what could possibly go wrong? Will Android soon be talking to the sky? What's up with the trouble SanDisk and Western Digital are in over their SSDs? Are children still being tracked on YouTube's "made for kids" channels? Has cryptocurrency become any safer, and what dangers are posed by the use of multi-party wallets? Is FIDO2 ready with post-quantum crypto? What's the latest on HTTPS by Default? And after looking at some feedback from our terrific listeners, we're going to examine the nature of heuristic programming algorithms with a case study in what can go wrong.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-936.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-936-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. He's going to talk about the Linux distro that's being bought by private equity. Oh, no. We'll also talk about what's going on with those SanDisk and Western Digital SSDs and why you should not buy them. And then a fascinating case study of a problem Microsoft has, maybe you've heard about it, with setting server clocks to ridiculous values, and maybe why it's happening. Steve does a deep dive that's fascinating. All that and more coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 936, recorded Tuesday, August 22nd, 2023: When Heuristics Backfire.

It's time for Security Now!, the show where we cover the latest news from the security ecosphere with this guy right here, Mr. Steve Gibson of GRC.com. Hello, Steve.

Steve Gibson: Hello, Leo. Great to be with you as we embark on year 19...

Leo: Oh, my goodness.

Steve: ...of this illustrious podcast.

Leo: And as we learned last week, there may be a year 20 and even a 21.

Steve: I think that, you know, I was shedding a tear to myself...

Leo: For your free time?

Steve: That we would probably not be together when Unix time wrapped around.

Leo: 2038, huh?

Steve: Yeah, I thought, you know...

Leo: I don't know if I'll be here.

Steve: Maybe.

Leo: You will. Good. I'm glad to hear it. That's exciting, yes.

Steve: So, we're going to - today's episode is titled "When Heuristics Backfire," inspired by actually the Norwegian engineer who brought this back to everyone's attention after seven years of pain and anguish by sending me a tweet to that effect. But anyway, so but of course we're going to get to that. We're going to answer some questions first.

Which Linux distro is selling itself to private equity capital, and what could possibly go wrong with that plan? Will Android soon be talking to the sky? What's up with the trouble SanDisk and Western Digital are in over their SSDs? Are children still being tracked on YouTube's "made for kids" channels? Has cryptocurrency become any safer, and what dangers are posed by the use of multi-party wallets? Is FIDO2 ready with post-quantum crypto? What's the latest on HTTPS by Default?

And, after looking at some feedback from our terrific listeners, which will generate some additional commentary as opposed to like one word, oh, that was a good idea - we do have some of those, though - but we're going to examine the nature of heuristic programming algorithms with a case study in what can go wrong.

Leo: What could possibly, you didn't say "possibly." And by the way, I'm just doing a little research on Wolfram Alpha of how many weeks from today to the end of the Unix Epoch.

Steve: Ah.

Leo: And I think he got it wrong. I think Wolfram Alpha thinks that the end of the Unix Epoch is 53 years from now, which is outright no.

Steve: You are exactly right. It's 2038. The geeks are well aware.

Leo: Yeah, yeah, yeah. So let me just - what's the date? Is it December 31st, 2038?

Steve: You could add 4.3 billion to January 1st, 1970.

Leo: Well, let's say it's roughly 801 weeks. So we will be at Episode 1737.

Steve: Oh, piece of cake.

Leo: Easy.

Steve: We're already halfway there.

Leo: Yeah. Think of it that way. We're halfway there.

Steve: We've already passed - we're over the hump.

Leo: So, yeah, starting counting. 15 years, four months, nine days. And, you know what, if you decided at any point, maybe even today, that that would be the last episode, at the end of the Unix Epoch we'll stop.

Steve: Honey, I have a USB cord. All I do is pull it, and that's it. I'm gone.

Leo: The ripcord. It's gone. He's dead. He's history. Well, anyway, we'll endeavor to make it another 15 years. That would be good. That would be good.

Steve: Yeah. I think that's, you know, as our illustrious William Shatner said, "Don't die."

Leo: Just don't die.

Steve: It's not a problem.

Leo: That's both of our jobs now for the next 15 years.

Steve: Don't die.

Leo: Do not die.

Steve: And I love the word because it would be see everything; right?

Leo: Right. Well, it comes from - was it Bessemer's conception of a prison, the panopticon, where prisoners would be under surveillance, but they wouldn't know. They knew that they could be, but they wouldn't know when the jailers were

watching them. So that the jailers would not have to watch all the time because the prisoners just - they wouldn't know. They'd just think we could be watching. And we all live in a panopticon now, actually, come to think of it. That's a depressing thought. All right. Picture of the Week time, Steve.

Steve: So this was one I had in the pile in reserve. And I liked it, and I think the caption is perfect for it. It shows sort of an idyllic setting. We have a slow-moving river in the background, and a little grassy knoll sort of lawn thing in the front, and what looks like a gate from the 1400s. I mean, it's like out of time somehow, two stone pillars on either side of a gate. But there's no gateage going off in either direction. So it's just this gate segment; right? I mean, there's like nothing to keep you from going around. And we've of course had many of these similar gates that we've had fun with on the show in the past.

But in this case, we have four sheep. And for reasons that are not explained by the photo, they are stuck behind the gate. Actually, one of them is kind of peeking around, wondering if maybe it might be all right to proceed. The other three, no. They're just at the gate, waiting apparently for it to open, despite the fact that you could easily go around on either side. And of course I gave it the caption, "This Is Why They're Sheep."

Leo: Yes.

Steve: So yes, because they don't know any better, they're just like, oh, the gate's closed. Damn.

Leo: Love it. See, this explains all the previous gate-without-wall pictures you've had. They're for sheep.

Steve: Yeah. They were, exactly, the sheep had apparently not yet arrived. We know had they arrived they could not have left. So they would still be there waiting for the gate to open. But, or maybe some nice person would come along and say, oh, you poor sheep.

Leo: Aw.

Steve: We'll let you through.

Leo: And clearly this one sticking his head around the side is the troublemaker, the black sheep.

Steve: Yeah, exactly, he's looking around, eh, why are we standing here on this side of the gate that has no sides? Don't know.

Anyway, I wanted to take just a moment of everyone's time to thank the many listeners who let me know after last week's podcast how glad they were to learn that I would not be retiring from this weekly communication after podcast 999. It had become clear to me that my plan to quit, for no particular reason, after 999 would have felt very weird as

that day was approaching; right? I mean, it would have been, like, okay, uh. But anyway...

Leo: Sad. It would have been sad.

Steve: Now we know that's not going to happen. So yay, and thank you, everyone, for sharing your sentiments with me.

Leo: Yeah. We're so happy.

Steve: I'm not certain whether it will matter to anyone here, but I wanted to note that the currently public company SUSE, you know, spelled S-U-S-E, the company behind the OpenSUSE Linux distro, has announced its plans to delist itself from the Frankfurt stock exchange and allow itself to be purchased and taken over by a private equity firm.

Leo: Oh, god, no.

Steve: I know. Their announcement read, in part: "EQT Private Equity has underscored its commitment to supporting the company strategically and financially, and to cooperate closely with SUSE's CEO and its leadership team." Dirk-Peter van Leeuwen, CEO of SUSE, said: "I believe in the strategic opportunity of taking the company private. It gives us the right setting to grow the business and deliver on our strategy with the new leadership team in place."

Leo: Oh, no.

Steve: Uh-huh. "EQT Private Equity's and SUSE's partnership," he says, "in a private setting has been fruitful before, and we're excited about the long-term potential of the company and our continued collaboration."

Right. Now, the whole truth is, as you've already been bemoaning properly, Leo, it's unclear what this means for the future of OpenSUSE. In general, as we've witnessed a number of times in our own PC and security industry, private equity firms do not purchase tech companies out of their love of tech. They typically purchase them because they perceive pockets of unexploited profit potential and some means of squeezing money in the short term from the golden goose. This is typically done through aggressive layoffs, cost cutting, selling off the less profitable pieces, and in general squeezing the remaining life - and the future - out of their new acquisition.

Users who have made their own investments in the future of such enterprises, like by adopting OpenSUSE as their Linux, should be forewarned that, if nothing else, change is coming. We don't know what kind of change, but it may not be good. So anyway, I just thought I wanted to let everyone know, OpenSUSE may not be quite so open in the future.

Leo: I mean, Novell's owned SUSE forever. So, I mean, but Open has always been the open version.

Steve: Yeah, yeah, yeah.

Leo: That's terrible.

Steve: Yeah. Okay. If imitation is the strongest form of flattery, it appears that Google is enamored of the satellite communications capabilities that Apple added to iOS 14. It appears that Google is testing a similar new Android feature that will allow its users to send SOS messages via a satellite connection in the case of an emergency. Now, this nascent feature was spotted in the code for Android 14, which is slated for release, like, any moment, like within days. It's supposed to be the end of August, and we don't have much August left. The feature is unannounced, and we should not expect to see it going live anytime soon.

According to Android OS expert Mishaal Rahman, the feature is not part of the main Android 14 OS and is just tagged as "demo code" in the source. So, you know, it's going to need accompanying radio hardware, I'd imagine, in order to talk to satellites. But anyway, Android users can probably assume that a feature similar to Apple's will be arriving at some point.

I should mention, in case any of our listeners might be impacted by this news, that SanDisk, my own preferred manufacturer of solid-state mass storage, and Leo...

Leo: Not anymore. Not anymore. Not anymore.

Steve: Yeah, I know, I know. SanDisk and Western Digital are both currently in the dog house over what is alleged to be their willing and knowing sale of defective SSD products. And not just a little bit defective. The allegations are disturbing. Here's the start of what Ars Technica had to say about this.

Ars wrote: "Amid ongoing pressure to address claims that its SanDisk Extreme SSDs are still erasing data and becoming unmountable despite a firmware fix, Western Digital is facing a lawsuit over its storage drives. A lawsuit filed last Wednesday accuses the company of knowingly selling defective SSDs. Western Digital branded SanDisk series of Extreme V2 and Extreme Pro V2 portable SSDs, and they're often being recommended, or were, by tech review sites." Ars wrote: "If you've considered a portable drive, it's likely you've come across the popular series in your search." And of course I'm wanting to let our listeners know because they are popular, and they may not have heard about this yet.

So Ars wrote: "Numerous owners of the drives, including Ars Technica's own Lee Hutchinson, encountered a problem where the drives seemingly erased data and became unreadable. Lee saw two drives fill approximately halfway before showing read and write errors. Disconnecting and reconnecting showed the drive was unformatted and empty. Wiping and formatting did not resolve things."

Leo: Oh, really. You mean your data didn't magically come back when you wiped and formatted, huh.

Steve: Wow, yeah. "Complaints about the drives littered SanDisk's forums and Reddit" - and Ars then provided four example links of typical postings - "for at least four months before Western Digital released a firmware fix in late May. The page for the update

claims products currently shipping are not affected. But the company never noted customers' lost data claims. It did, however, name the affected drives."

So we have the SanDisk Extreme Portable 4TB, the Extreme Pro Portable 4TB, same thing in 2TB and 1TB, that is, the SanDisk Extreme Pro Portage. And also the Western Digital My Passport 4TB. And then they finish: "Subsequent reports from The Verge, which received a replacement SSD, and some Reddit users, though, claimed the drives were still broken. Western Digital didn't answer requests for comment about newfound grievances."

Okay. So for what it's worth, I just wanted to inform our listeners that this has been and apparently still is a problem. And if I were storing important data on any of those named drives, I'd be a little nervous. I wouldn't maybe store anymore data on them.

Now, hearing that - and I actually have sort of maybe an explanation of why. Hearing that the drives started having trouble when they became around half full was interesting to me. As an engineer, I have been astounded and shaking my head when I learn the lengths to which today's SSDs are going to achieve the storage densities they offer.

For example, get a load of this: Remember first that NAND-style SSD storage cells, which is what we are using, the storage cell is really just a capacitor. It's an itty-bitty - and yes, that's an engineering term - it's an itty-bitty spec of metal sitting on top of an insulating coating. The act of writing to that cell involves cranking up the voltage internally to deliberately break down that layer of insulation to inject and strand electrons on that itty-bitty island of metal. Once there, their presence can be sensed thanks to the magic of field-effect transistors. But the point is, the number of electrons which have been stranded on that electrostatic island is an inherently analog quantity.

Now, the designers of the earliest FLASH memory, which regarded the fact that all of this worked at all to be a miracle, were quite content to only either fully fill or fully empty the islands in order to store zeroes and ones there. But then management came along and demanded more density. The engineers explained that they had already made the electrostatic islands as itty-bitty as possible. But then one of them made the mistake of suggesting during a brainstorming meeting that they might be able to store two bits of data on each island by storing varying amounts of electrons, thus doubling the effective storage density of the same number of itty-bitty memory cells.

Instead of storing a boring old zero or one, it would be possible to store a zero, a one third, a two thirds, or a one. Since this gives us four separate storage levels, that's two bits of data. Whereas the original NAND memory cells were known as SLC for single-level cell, this next generation were MLC for multi-level cell. And, of course, once you've opened Pandora's Box of storing what are essentially analog levels of electrostatic charge in cells, why stop at two bits? How about three or four?

Okay. So with this bit of background, here's what I learned that made me shake my head. The SSD manufacturers are aware that storing fewer bits per cell is better. It's faster to read and write, and it's more reliable. So believe it or not, today's most advanced SSDs start out by storing fewer bits per cell because it's faster and more reliable. It makes their product benchmark faster when it's mostly empty.

But as their users continue to fill them up with data, the SSD begins to run out of storage at that lower, faster, and safer density. At some point of fullness it needs to switch over to storing more bits per cell in order to actually deliver its promised and rated full storage capacity. To do that, it starts reading and rewriting existing data which was stored at the lower bit density into higher bit density. Essentially, it's on-the-fly cramming more bits into fewer cells because, as it turns out, its owner actually wants to use the storage capacity that they purchased.

The extra complexity that had to be built into the controller to track and accomplish all of this is somewhat mindboggling, but it's obviously worth it to its product's developers. The flakiness inherent in how far SSD storage density has been pushed is one of the reasons I became certain that a continuing investment in SpinRite would be warranted into the future.

Okay, now, only SanDisk's engineers know why, when their drives are storing about half of their rated capacity, they suddenly developed a problem that crashed the entire drive. It could be that the drive had actually filled itself with data stored at half-density, and now needed to get serious about doubling-up on density; and that some flaw in the drive's firmware was then triggered. We'll likely never know. This brings to mind something else that happened and was discovered last week by one of SpinRite's testers on one of their thumb drives. But I've already taken up a lot of time on this, so I'm going to share that interesting story about something else that happened and was really interesting next week.

Leo: Whew. Wow. By the way, just to correct the record, I just looked through my Amazon orders. I've never bought a SanDisk SSD. I always do the Samsung EVOs.

Steve: Oh, you're right, Samsung is also what I was thinking.

Leo: Yeah. We like the Samsungs.

Steve: Yeah, yeah, yeah.

Leo: I don't have any SanDisks. SanDisk I buy their SD cards, their solid-state memory cards, yeah.

Steve: Yes, and I have a bunch of SanDisk thumb drives because I like those.

Leo: Yes, and I do, too.

Steve: It's the Samsung NVME, beautiful memory.

Leo: Or even like the 2.5-inch SSDs, yeah.

Steve: Or even the big, yeah, SSDs, yes. Good. I'm glad you [crosstalk] that.

Leo: Samsung is not SanDisk, thank goodness.

Steve: Right, right. Okay. I ran across a blurb of news that read: "YouTube children's privacy: An Adalytics report found that advertisers are still tracking viewers of videos made for kids, despite a 2019 promise from YouTube to stop delivering personalized ads on these types of videos. And senators are now seeking a formal inquiry into the company for breaching its U.S. COPPA (C-O-P-P-A) laws."

COPPA stands for Children's Online Privacy Protection Act. And so I was curious about this, so I followed the Adalytics link. Adalytics appears to be a scrupulously neutral adtech industry watchdog. They make a point of saying they don't take, they do not accept any money from the adtech industry because they don't feel they could do so without an inherent conflict of interest, which I think is obviously true. Their report is long, but I'll share just the top few takeaways which will give everyone the gist.

They said: "First, YouTube's CEO said in 2019 that the platform would 'limit data collection and use online videos made for kids only to what is needed to support the operation of the service." So, which, you know, is like being logged on, I suppose. "However, YouTube appears to be setting or transmitting 'advertising' cookies and identifiers on the devices of viewers who are watching 'made for kids' videos as of last month, July 2023." So that promise was not kept.

"Second, YouTube's CEO said in 2019 that the platform would stop serving personalized ads on 'made for kids' content. However, demographically and behaviorally personalized ad campaigns appear to have ads being served on 'made for kids' YouTube channels as of last month, July 2023.

"YouTube is serving ads from many 'adult' Fortune 500 advertisers and major media agencies on YouTube channels that are labeled as 'made for kids.' These include major brands such as Mars, Procter & Gamble, Ford, Colgate-Palmolive, Samsung, and many others. In some 'adult' brands personalized ad campaigns, the top YouTube channels by clicks or click-through rate are popular 'made for kids' YouTube channels." In other words, kids are clicking on these things. And get this: "Such as 'ChuChu TV Nursery Rhymes & Kids Songs,' 'CoComelon Nursery Rhymes & Kids Songs,' or 'Kids Diana Show.'" Click-throughs are happening for the ads that are on those, and they're going to adult brands, not kids' brands.

"The viewers of 'made for kids' YouTube videos appear to be clicking on ads and brands websites such as Michigan State Police, Disney, BMW, Hyundai, and Verizon. And those companies are harvesting and sharing metadata on those viewers with dozens of data brokers upon click-through. This raises the possibility that brands have 'data poisoned' their first-party datasets with data derived from thousands of viewers of 'made for kids' videos."

Anyway, all of this sounds like YouTube is absolutely not taking the requirements of COPPA for granted, I mean COPPA seriously. They are, you know, just like blowing it off. Anyway, the report goes on and on like that. And this is clearly not what the COPPA act had in mind. So it's hardly surprising that the U.S. Senate, which has been all worked up over TikTok recently, as we know, is wondering what's going on with this popular U.S.-owned property? In the show notes I titled this story "You asked for it" since that's clearly what Google is doing. You know? I mean, they're just blatantly ignoring this. This entire tracking and data-brokering industry seems to me needs a flushing, and at this rate that might not be too far away.

Since we were just talking about the danger of losing access to an email forwarding service, I got a kick out of this little bit of news: Apparently the operators of the 8Base ransomware operation lost all of the data that they had previously stolen from their victims when the AnonFiles cloud hosting file storage service closed down without notice last week.

Leo: Aww. Aww.

Steve: And I saw elsewhere the announcement from AnonFiles. They just said: "We are so disgusted, and we are so fed up with trying to offer, like trying to do the right thing, and offer for the value and for the sake of the Internet an anonymous file hosting service. We are being overrun with malicious use of our anonymous files hosting." So they pulled the plug, and with it they took a bunch of data that had been exfiltrated from some victims of the 8Base ransomware group. So yay for that.

Okay. Meanwhile, the blockchain security firm CertiK, who often provides interesting tidbits that we talk about here, says that a group of Canadian scammers are responsible for stealing millions of dollars during the past few years. The group's members operate by hacking the Discord servers used by cryptocurrency communities and posting phishing links to hijack their users' wallets. CertiK claims to have identified the real-world identity of one of the group's members. Going by the pseudonym of "Faint," (F-A-I-N-T), the individual is believed to have stolen more than $1 million worth of assets since late 2022. Faint is the group's second publicly-identified member after another blockchain investigator exposed a hacker named Soup last month.

So anyway, I just tossed that one out there to remind everyone that nothing much appears to have changed in the cryptosphere of late. It remains a wildly immature and insecure mess still.

In one other bit of crypto news, there's another example of this fundamental immaturity within the entire industry. Major crypto-industry players jumped onto an unproven and not fully tested algorithm due to the promise of what it offered. And of course isn't that the story of this entire cryptocurrency phenomenon? Earlier this month, during Black Hat, researchers from Verichains disclosed three vulnerabilities in a cryptographic protocol called TSS which is used to create multi-party crypto wallets known as MPCs. Okay. Here's how Binance, one of the prominent users of this new TSS algorithm, describes it on their site. TSS is short for Threshold Signature Scheme.

They said: "Threshold Signature Scheme (TSS) is a cryptographic primitive for distributed key generation and signing. The use of TSS in blockchain clients is a new paradigm that can provide numerous benefits, especially in terms of security." Whoops. "In the broader sense," they wrote, "TSS can influence the design of key management systems such as crypto wallets, and lead the way for native support in DeFi use cases. Having said that," they wrote, "TSS is still a new technology, so the risks and limitations should be considered."

Unfortunately, apparently they didn't take their own advice. Or if they did, they didn't consider them enough because the last part they got right.

The Verichains researchers named their attacks on TSS "TSSHOCK". TSSHOCK exploits vulnerabilities in some of these MPC, this multi-party crypto wallet technology, through their implementation of the threshold elliptic curve digital signature algorithm (ECDSA). And so this is a new thresholding style. The exploit of TSSHOCK can allow threat actors to steal cryptocurrency from individual users or major institutions while leaving no trace of the attack on the client site.

And here's the good part: Major companies like Binance, Zengo, Multichain, THORChain, and ING Bank use exactly these vulnerable threshold elliptic curve DSA software implementations. And on top of that, TSSHOCK is the second major multi-party crypto wallet vulnerability disclosed this month, the first being BitForge. And Leo, after taking our second break, I'm going to tell everybody what happened earlier this month with BitForge.

Leo: Oh, lord. I bet it's good. Can't wait. Yeah, it's always fascinating. By the way, I did a little research. The Unix Epoch ends when it crosses over the 32-bit threshold on January 19th. So...

Steve: Early 2023 [sic].

Leo: Early 2023 [sic].

Steve: In which case I'm sure we'll still be here.

Leo: We only have 752 more episodes after this.

Steve: Yeah, like I said, we're like way past the halfway point.

Leo: So I just wanted to give you a heads-up, you know.

Steve: I'm feeling great, Leo. I hope you're doing great.

Leo: It's only 14 years. I think I can make - I'll be 80. That's not so bad.

Steve: You know, our listeners who were having their first children born at the beginning of the podcast will be grandparents by the time...

Leo: You know what, let's put a pin in it. I'm planning on it.

Steve: Okay.

Leo: June - because we could then talk about, you know, the new Y2K problem, the Y238 problem.

Steve: That's right.

Leo: And then wrap things up. January 19th, which is a Tuesday, by the way.

Steve: Perfect.

Leo: 2038.

Steve: Perfect. I like...

Leo: We don't even have to do a special.

Steve: So for the sake of painting a full picture of just how rocky things still are in the crypto industry, I should also touch on this earlier discovery which I referred to, BitForge, which is made by a group known as Fireblocks. Two weeks ago, on August 9th, Fireblocks posted their discovery under the title "Fireblocks Researchers Uncover Vulnerabilities Impacting Dozens of Major Wallet Providers."

And they explained in their posting: "Today, Fireblocks Cryptography Research Team announced the discovery of multiple zero-day vulnerabilities in some of the most used cryptographic multi-party computation (MPC) protocols, including GG-18, GG-20, and implementations of Lindell 17. If left unremediated, the exposures would allow attackers and malicious insiders to drain funds from the wallets of millions of retail and institutional customers in seconds, with no knowledge to the user or vendor.

"The series of vulnerabilities, dubbed BitForge, impact popular wallet providers like Coinbase WaaS." I guess that's Wallet as a Service, to which you have to ask, what could possibly go wrong with that? We also have Zengo and Binance using this. They said: "Following industry-standard 90-day responsible disclosure processes, Coinbase WaaS and Zengo have since fixed and resolved the identified issues." They don't mention Binance there. They said: "In addition, the academic papers which had the details of their flaws redacted have been revised. The Fireblocks Cryptography Research Team findings were presented during the Black Hat USA conference on Wednesday, August 9th, and will be shared at Defcon on Thursday the 10th."

Pavel Berengoltz, Co-founder and Chief Technology Officer at Fireblocks, said: "As decentralized finance and Web3 continue to gain popularity, the need for secure wallet and key management providers is evident. While we are encouraged to see that MPC" - that's the multi-party stuff - "is now ubiquitous" - wow, I'm not encouraged, but okay - "is now ubiquitous within the digital asset industry, it's evident from our findings, and our subsequent disclosure process, that not all MPC developers and teams are created equal. Companies leveraging Web3 technology should work closely with security experts with the know-how and resources to stay ahead of and mitigate vulnerabilities. Maintaining and updating core infrastructure technologies, like Web3 wallets, is crucial" - get this - "in preventing thefts and attacks, which amounted to" - ready for this - "nearly $500 million in just the first half" - just the first half, Leo, of this past year 2023.

Leo: Half billion here, half billion there, it starts to add up.

Steve: Yeah. And where is this, like who has all this money that's being lost?

Leo: Well, they don't anymore.

Steve: It's just an astonishing, astonishing amount of money is being drained out of people. It's like, wow. Anyway, "Wallet as a Service."

Leo: No. Wrong. Yeah, I'll tell you what the service is.

Steve: Yeah. Wow.

Leo: We'll take your money. Then we'll take your money.

Steve: Wallet as a drain plug.

Leo: Yeah.

Steve: Anyway, from the beginning of this cryptocurrency odyssey, our one constant piece of advice here has been to keep your wallet safely offline.

Leo: And then don't forget your password. Whatever you do.

Steve: Do not format the hard disk where you had your 50 bitcoins.

Leo: That's right.

Steve: Boy, that was a dumb one.

Leo: Don't even - just forget that ever happened.

Steve: Anyway, enough said.

Leo: Enough said.

Steve: If you want to play with crypto, just please be careful and heed the age-old investment advice to never gamble more than you're able to lose.

Leo: See, we - this was - we didn't gamble anything. It cost you nothing to get that 50 bitcoin.

Steve: Only heartache after I formatted the hard drive.

Leo: Yeah, I never have purchased bitcoin. It was all donated. So we didn't gamble anything. But people were buying bitcoin. Not now, I hope not, anyway.

Steve: Well, and, you know, in all fairness, when I was in Boston with you, Leo, when we did the event there and had a whole bunch of our podcast listeners show up, we made some bitcoin millionaires.

Leo: What?

Steve: There was a guy came up and said, "Thank you, Steve. I listened to your bitcoin podcast. I got involved. I'm now retired. I've never had to work again."

Leo: Holy cow.

Steve: And this was a young guy.

Leo: Nice, very nice.

Steve: Yeah. So that was very cool.

Leo: Yeah.

Steve: Unfortunately, I didn't follow my own advice.

Leo: Oops. Oh, god.

Steve: Okay. So does anyone want a "quantum resilient" FIDO2 security key?

Leo: Yes.

Steve: Well, it may be of more interest once it's possible to use Passkeys widely. But in any event, Google has developed the first-ever version of a FIDO2 security key that includes protections against quantum computing attacks.

Leo: Wow. I didn't know we'd had a way to do that.

Steve: Yeah. Well, remember we have those nascent algorithms.

Leo: Yeah.

Steve: Like Dilithium. And of course, yeah, because of quantum. And they exist. So the implementation comes from a new version of what's known as OpenSK, as in Open Security Key, which is an open-source project on GitHub that provides firmware for security keys. Google says this new OpenSK firmware version uses a novel ECC [Elliptic Curve Crypto] Dilithium hybrid signature scheme its engineers developed along with academics from ETH Zurich. The project doesn't yet appear to be, like, to be production ready. It's definitely still experimental and not ready for prime time. But it appears that when quantum computers eventually happen, we'll be ready, even if there still aren't many sites supporting Passkeys. At least those that do, you'll be safe.

And actually, this is important for Passkeys because it uses public key crypto, which is the form of crypto which quantum computers are expected to crack. Symmetric crypto,

that's not quantum unsafe. So all of the symmetric crypto that we have, you know, that's fine. It doesn't need to be changed. It's the public key crypto that is the problem. So, you know, it's nice that we'll be ahead of the game.

Chrome 116, which is today's current release, contains a not-enabled-by-default feature which is scheduled to be enabled by default in Chrome 117 at the start of September, so a couple weeks from now. The feature is called "safety check extensions," which will allow Google to display notifications to Chrome users when one or more browser extensions that they currently have installed and running are removed for some reason from the official Web Store. The notifications will be shown when an extension is marked as malware, the extension is removed for Terms of Service violations, or when the extension is unpublished by its developer.

Now, that seems like an uncontroversial good thing, though it's a bit surprising that Chrome doesn't already do that. But certainly good that they're going to be adding that. It's in there now. You can turn it on. But it's not on by default. It'll be on by default with the next release of Chrome.

And speaking of defaults, even with TLS certificates now being free, we're still not all onboard the HTTPS train. So last Wednesday, in a posting titled "Towards HTTPS by default," Google announced their next move. They wrote: "For the past several years, more than 90% of Chrome's users' navigations have been to HTTPS sites, across all major platforms. Thankfully, that means that most traffic is encrypted and authenticated, and thus safe from network attackers. However," they wrote, "a stubborn 5 to 10% of traffic has remained on HTTP, allowing attackers to eavesdrop on or change that data."

Okay. Now, just wait a minute. It's not "stubborn," Google. It's just ignoring you. It doesn't care. Imagine that. Now, okay. As we know, this is certainly traffic that really doesn't need any security of any kind and is 100% completely happy being unauthenticated and out in the open. But it apparently annoys Google nevertheless. I have an example in the show notes of exactly such a site. I refer to it constantly while working on SpinRite since it contains a comprehensive list of every PC interrupt from 00 to FF, all of their sub-functions and their arguments. It is wonderful, and I'm very thankful for it. I've copied the entire site against the inevitable day when it finally disappears. It's http://www.ctyme.com/intr/int.htm. And yes, it is proudly HTTP. And I'm pretty sure that it's never going to change. As an aside, it also makes my own website look quite modern by comparison. It even has the old mailbox icon for email.

Anyway, Google is still annoyed, so here's what they're going to do. They said: "Chrome shows a warning in the address bar when a connection to a site is not secure, but we believe this is insufficient. Not only do many people not notice that warning, but by the time someone notices that warning, the damage may already be done." Right. You know, what? The damage from looking up some interrupts in a table. Anyway, maybe some dastardly Russian is going to do that to subvert SpinRite. I don't know.

Leo: We had this argument, not argument, discussion about my site where we don't have any logins, like...

Steve: Exactly. Exactly.

Leo: Why does it need to be secure?

Steve: Exactly.

Leo: Yeah. I mean, you can make an argument that it could be impersonated. But then, still, what are they going to get? There's no login.

Steve: It's just, yeah, exactly, it's just some data that someone wants to share on the Internet. That's it. It's some data that somebody wants to share on the Internet.

Leo: The irony of this interrupt jump table is it's hosted by Computer Tyme, which is a web hosting company that is also not HTTPS.

Steve: And they're apparently, now, one can't really discern from that...

Leo: Look at that, there's John C. Dvorak, and I get no spam. This has got to be just a site that's been around for a while, and it's dead; yes?

Steve: I think you just scrolled past an Apache icon on the left.

Leo: Oh, my god, this thing is ancient. Uh, wow. Computer Tyme. I think this is the guy who used to provide Dvorak with his spam filtering. Which is wild. Outbound spam filtering. So it's just some guy. Yeah, he says, if you saw the John C. Dvorak article, "The Death of Email," in PC Magazine, or Show 18 on This Week in Tech. Like this is literally 15 years ago. We can provide you with the individual Marxmail accounts you want. We do host John's email. I wonder. I wonder if it's still - anyway, very cute. I guess he doesn't have time for HTTPS.

Steve: So it's a perfect example of a site that's like...

Leo: It's doesn't need it. Doesn't need it.

Steve: Yeah, stop bugging me, I just want to put some stuff on the Internet.

Leo: His spam filter, by the way, is HTTPS. So he knows enough to do it when it's needed.

Steve: Right.

Leo: Yeah.

Steve: Yeah, exactly. And that's, you know, old school.

Leo: Old school.

Steve: I would argue it's just fine. Anyway, Google says the damage may already be done if you go to a site that isn't HTTPS.

Leo: So there.

Steve: Literally. "We believe," they say, they wrote, "We believe the web should be secure by default." That's right. No insecure traffic allowed. Wow, you know, you'd think they were selling certificates. But they're not. I think they're getting a little far over their skis with this thing. But after all, they're just a web browser; right? Anyway. So here it is. "HTTPS-First Mode" - that's what they're calling this new thing. "HTTPS-First Mode lets Chrome deliver on exactly that promise" - no insecure interrupt tables - "by getting explicit permission from you before connecting to a site insecurely." Oh, great. So anyway, thank goodness I'm over here on Firefox, where that's not happening.

They said: "Our goal is to eventually enable this mode for everyone by default. While the web isn't quite ready to universally enable HTTPS-First Mode today, we're announcing several important stepping stones toward that goal. Chrome will automatically upgrade all http:// navigations to https://, even when you click on a link that explicitly declares http://."

They said: "This works very similarly to HSTS upgrading, but Chrome will detect when those upgrades fail, for example, due to a site providing an invalid certificate or returning an HTTP 404, and will automatically fall back to the http:// that you originally asked for. This change ensures that Chrome only ever uses insecure" - and I would argue not secured, but no - "insecure HTTP when HTTPS isn't truly available, and not because you clicked on an out-of-date insecure link. We're currently experimenting," they said, "with this change in Chrome 115, working to standardize the behavior across the web, and plan to roll out this feature to everyone soon. While this change cannot protect against active network attackers" - you know, fine - "it's a stepping stone towards HTTPS-First mode for everyone, and protects more traffic from passive network eavesdroppers." Again, for sites nobody gives a crap about.

Okay. So all that said, overall, this is a good thing. It would be annoying to need to push past another warning screen before being allowed to access an HTTP site. That does seem like overdoing it. But certainly trying first to access a URL over port 443, bring up a TLS handshake, and see whether that http:// URL is also available via HTTPS, that makes sense.

However, it is worth noting that technically HTTP URLs and HTTPS URLs are actually referring to different resources. Nothing anywhere ever says that the URLs of those differing protocols always or ever needs to or should refer to the same web resources. It's true that they generally do. Most web servers serve the same content regardless of whether their visitors come in via HTTP or HTTPS. And these days, most attempts to come in via HTTP are immediately redirected to the same URL over HTTPS to, like, fix the link.

But that doesn't necessarily need to be so. It's just the way things have typically evolved over time. So I can see why Google is being cautious. They clearly want to kill HTTP entirely, and I'll bet they even want to remove its support from Chrome. But it doesn't look like it's going to go away quietly. So it's putting up a bit of a fight.

One last bit before we get to our listener feedback. I wanted to note that WinRAR is now at v6.23. I've long been a fan, and I'm a registered owner of WinRAR. So I wanted to note that a recent update to v6.23 earlier this month closed a couple of maliciously exploitable holes that attackers could use in a targeted attack to run their own code on

their victim's system. So if you are, like I am, a WinRAR user, you want to probably go over there and grab it. I looked. I fired up my copy. Nowhere is there an update, check for update, check for new version or anything like that. There is a link to go to their website automatically, but they make you do that.

And actually I'm not sure that's a bad thing, although it would be nice to be notified if like an important update were made available. This actually happened earlier this month, you know, weeks ago, and I didn't know about it until I ran across a mention of it elsewhere. So anyway, now all of our listeners know.

Leo: That's why you have to listen. We're going to tell you.

Steve: That's exactly right. That's exactly right. Until 2032, when the Internet will finally end.

Leo: 2038, yes. Don't rush me. Don't rush me.

Steve: Oh, no, no, no. That's right. Got to wait to the bitter end.

So iam-py-test, that's his Twitter handle, he said: "Hello. Hope your weekend is going well. Relating to this week's episode, I thought you might find it interesting to know uBlock Origin disables Topics by default."

Leo: Oh, interesting.

Steve: Uh-huh.

Leo: You can block it.

Steve: Yes, by modifying the Permissions Policy header. "Thanks for the podcast and have a nice day." So as we mentioned last week, there is - again, Google is wanting to be really, you know, really kind and gentle about this. It is possible for a website to add a header that turns off Topics completely and so that every time the web browser receives any content from a site, if that header, there's a permissions policy header says no Topics, then they're disabled for that site.

And I suppose, you know, we should have expected this from Gorhill; right? I mean, he makes Dvorak seem all giddy.

Leo: He's a cranky fella, he is.

Steve: Yeah, yeah, he really is. In the GitHub dialog on this, Gorhill quoted a comment from someone else which read - and he was quoting stephenhawk8054, who said: "Gave more thoughts to this, and I think we should add it to uBlock filters - Privacy list. I don't think a user interface component for this is the way to go, given that so far it's an API supported only by one browser, and that the API is to serve advertisers in the first place."

Okay. Now, as we know, Topics is a new component of the underlying Chromium browser core. All of the many Chromium-based web browsers will at least have the opportunity to support it easily, if they don't deliberately block it. And initially, you know, some of those may choose to block it...

Leo: I'm sure Brave will, and yeah, yeah.

Steve: ...just because they - yes, exactly.

Leo: Privacy-focused ones, yeah.

Steve: Right. And, you know, this is the sort of major change that can be expected to take time, and will require a great deal of re-education, which we, at least, began on this podcast last week. Just look. Google's like all pissed off that HTTP still works. And that's, you know, good luck on killing that off.

Leo: Takes a while.

Steve: Does take a while. Everyone here enjoys at least knowing where the bleeding edge is, and we always find it. So some patience will be needed. The EFF will likely never come around to the idea of helping websites to monetize their visitors by providing advertisers with any information at all about who's visiting their pages. You know, and who knows, maybe they're right. It might be that all this profiling which grew out of tracking which was itself an unintended side effect of cookies has always been something that data brokers have just been getting away with only until the much slower-moving government regulations finally catch up. You know? Perhaps absolute anonymity is what we will eventually get, thanks to legal frameworks rather than technological frameworks. So anyway, no matter what happens, we're going to have fun following it here on the podcast.

Leo: There's a whitepaper which you probably read from Mozilla about this. And their complaint, we've talked about it since your show last week on several of our shows, their complaint was that it's possible to deanonymize this topic information. If you visit enough sites, you know, it's possible to deanonymize it. And that's of course always the concern. If you can keep it anonymous I think it's...

Steve: I didn't see a whitepaper. I did not see the whitepaper. I will look at it. I don't know how that's possible, so I'm wanting to find out.

Leo: Yeah. Well, you know, they may be making the assumption, which isn't probably wrong, that deanonymization is easier in many cases that one would expect, you know. We've seen a lot of cases where, wow, you can deanonymize...

Steve: I'm curious because there's a whole bunch of anti-deanonymization stuff in there.

Leo: No, I know, I know. Yeah.

Steve: Anyway, so if Topics were to succeed, and it got universal cross-browser adoption and industry endorsement, while Gorhill's uBlock Origin continues to block it, then we'll ask Gorhill for an easy-to-use exception switch, as he has done on his UI for other things, in order to turn it back on. Or if he refuses to do that, then he just may feel like, okay, blocking it is our job, then I'm sure it's possible to tweak the rules in order to allow it.

Leo: It may be moot. Has he said anything about Manifest 3? Because Google is planning to, I think, do what will essentially disable uBlock Origin in the long run.

Steve: Yup. You're right.

Leo: So it may be moot. You may, you know, as I have said and you have said, might be better just to use Mozilla and have done with it. I mean, that was Cory Doctorow's complaint is not so much Topics, but just that Google's attempt to really monopolize, they already monopolize online advertising, both as a buyer and seller. And they would love to monopolize browsers, as well. And that's, you know, that's a problem.

Steve: And I'm kind of feeling sad about this because I'm feeling less and less charitable toward Google.

Leo: Yes, I feel sad about it, as well. And I feel the same way.

Steve: You know?

Leo: Yup.

Steve: When I put a search into Google, I used to get a clean page of links.

Leo: Not anymore.

Steve: And now it's not that anymore.

Leo: Unh-unh.

Steve: It's really - it's really changed. So The Real Veran Dontic, he said: "Hi, Mr. Gibson. I love Security Now!. I noticed a slight logical flaw on SN-935 regarding the GPL. The GPL only requires that you make available the source code of your modified GPL code with the distribution of that code. Russian citizens could safely honor the GPL without actively contributing to open source."

Okay, now, first of all, it sounds like he's been listening to Ant with this Mr. Gibson business. I appreciate the gesture of respect. But please, everyone, we've been together

for 18 years. I'm just Steve. And I do appreciate Veran's clarification about the GPL after I fumbled my mention of it last week. But even so, assuming that Russia invests in modifying Linux to suit their own purposes and thus acquires an inherent feeling of propriety over "their result," who would imagine that Vladimir Putin's minions will have any interest in providing their citizens with the source code, any more than Microsoft does with Windows; or that they would make it available to anyone, including the Linux Project, in any form or fashion? What benefit would Russia accrue from doing so?

So I'd be willing to bet that what we're going to see here is the Linux source forked and never shared again. I would be surprised if anything else happened. But anyway, you know, thank you, Veran, for the clarification.

Emma Sax tweeted: "I just finished listening to your latest podcast, and I've been doing some thinking on your concerns about an email forwarding service, like Fastmail, just choosing to shut down. I've come to the conclusion that that concern is no different from Gmail or Outlook shutting down their servers, which millions of people rely on. At any point, any common email providers could just choose to not host email anymore, and tons of people would be forced to change their emails all over.

"The only way to get around it is by running your own email server, which you mention. And most people aren't willing to go that far. So for most people, we might as well just accept that we're relying on other peoples' servers for things. And so just do your research to make sure you're choosing one that seems to be the most reliable to you, as in owned by a big company versus a single person, et cetera."

And thank you, Emma. I think you're exactly right. I think I'm sensitive to the fact that we do see services come and go, even from major providers. Mozilla once offered a terrific file transfer service which they, too, shut down because it was being abused to transfer malware. And we've seen several companies stop offering ephemeral credit card numbers, which was a useful service while it was there. The difference with those services is that they don't incur the same level of persistent dependence that an email forwarding service does when it's used to anchor online account recovery.

So Emma's right. I think that the optimal thing to do is to choose your email forwarding provider with your eyes open and with an awareness of the importance of any such service's continuing support and their existence going forward. And if you ever receive any notice of any pending service discontinuation, don't wait to migrate your existing forwarded accounts to some other provider. So, you know, just recognize the level and nature of that dependency.

AMSather, he said: "Hey, Steve. Topics sounds great, but what's to stop site operators from storing the Topics supplied to them and aggregating it to sell to advertisers?" He said: "Love the show. Longtime listener and SpinRite 6 owner. So happy you're not stopping after 999."

Okay. This is one of the trickier bits of the Topics API. And I do hope that its subtlety does not keep it from being appreciated. Remember that brain twisting part about which topics a requestor could receive from a browser? One of the things to appreciate about the browser's role in this - that is, in Topics - is just how much of the burden for enforcing the user's privacy has been placed on the browser. Virtually all of it. And because it's client-side, that's where its strength and privacy enforcement comes from. The browser, with Topics, needs to keep track of all kinds of information for its user. But fortunately, that's only needed on a moving four-week window, which is actually part of the clever tradeoff that's been made. It does not need to keep it longer than four weeks.

So in order to obtain those three interest Topics from a user's browser when the user is visiting a website, the requestor - who would typically be an advertiser serving ads

across the Internet - must have previously queried that user's browser at some site which the browser associates with each of the topics that would be returned to the requesting advertiser. And I know it's just like this is hard to put into words, so here's, again, an example.

Three topics will be chosen, one for each of the past three weeks, based upon the domain the user is visiting because of that hash function which chooses one topic from among the top five during each of the three past weeks. Okay. So say that one of the three selected topics which would be returned to someone requesting them on a given site, an advertiser, say that one of the three topics is "Fish." This would happen if the user's use of their web browser during the previous three weeks has temporarily taught their web browser, because of the web sites they've chosen to navigate to, that they currently have an interest in fish. So "Fish" would be among their current top interests.

But at this time they are not at a website that their web browser associates with fish. They're at a site that their web browser associates with gaming, for example. But nonetheless, an advertiser on that gaming site is asking for the three chosen topics of interest to this user now. And here's the crux: "Fish" will only be returned as one of the three topics to that advertiser if sometime during the preceding three weeks that same advertiser queried for this user's three topics at a site which the user's browser associates with fish."

In other words, in order to be told that this user at this gaming site has an interest in fish, that same advertiser needs to have recently previously encountered this user's browser at a website that the browser associates with fish. And unless that's true, "Fish" will be redacted. It'll just be eliminated from the three topics that will be returned to the user, and nothing will be substituted in its place. The advertiser just gets less information. So anyway, there's just no easy way to explain this, but that's it again.

Okay. But so what does that mean? Because now back to our listener's question, what does that mean? He said: "What's to stop advertisers from storing the topics supplied to them and aggregating them to sell to advertisers?" And that's part of the subtle beauty of this system. Ask yourself what topics a browser will return to any static website. Unlike advertisers, websites don't encounter their visitors' browsers at other websites. Only third-party advertisers have that sort of cross-website reach. Websites have a first-party relationship with their visitors. So they never see them anywhere else. That means that the tricky Topics filter will only ever return the same topics to a website that the browser associates with that website. This means that websites are unable to learn anything they didn't already know about their visitors.

And of course note that this makes it completely different from FLoC, which was blabbing with that cryptic hashed token to every site someone visited, whether or not they'd ever even been there before. Not so with Topics. Again, I'm a little worried because it is complicated. And, you know, the world has a hard time dealing with complicated things, especially when they're asked, you know, doesn't that sound great? They go, uh, what's for dinner? So anyway. Let's hope.

Robert Gauld, G-A-U-L-D. He said: "@SGgrc In SN-935, I think consecutive characters refers to sequences as in ABC, def, 567, et cetera."

Okay. So, yes. Thanks to Robert and many other listeners. Remember, Leo, that ridiculous set of password rules? And we had a lot of fun because they said no two consecutive characters. And I said, wait a minute. How can you - or was it never more than two consecutive characters? So how could you have a password greater than two characters if you can't have more than two that are consecutive. Anyway, he, Robert, and many of our listeners said, you know, Steve, I think they meant sequential.

Leo: Oh. Yeah, but they didn't say sequential, so...

Steve: They did not say sequential. So, you know, and there were several of those rules that were kind of fumbles. Remember, like, one was a proper subset of the other, so you didn't even need that second one because the first one, you know, was a full superset of what the second one was. I mean, that mess, that set of rules was a mess. But anyway, I wanted to thank all of our listeners who said, you know, I think they didn't mean consecutive, they meant sequential.

Leo: Yeah. Okay.

Steve: And finally, yeah, Rob Mitchell said: "You know what would be great? If you posted your Picture of the Week for each episode here on Twitter. You could include a link to each episode, so you'd also be doing some promotion, too. I rarely take the time to find the photo of the week, but I'd surely find it here."

And Rob, your request, and actually I saw it several times, it must have been in the wind somehow this past week because no one ever mentioned it before. But a bunch of people did this time, maybe because the pictures lately have been a little obscure, and they've required a great deal of explanation in order for me to do it over a non-visual medium. Anyway, from now on, and I already did it earlier today, the podcast Picture of the Week will accompany the link to the show notes on Twitter.

Leo: Now you're using Twitter right.

Steve: That's right.

Leo: Just as they change their name to X.

Steve: Speaking of which, Leo, he took TweetDeck away from me.

Leo: He did.

Steve: That [muttering].

Leo: Yeah, people are very upset about that, yeah.

Steve: Oh, my god. Using the normal interface is awful.

Leo: Yeah.

Steve: No, TweetDeck was a godsend. But on the other hand, I refuse to pay that SOB $89 a year or something? I'm not paying for this. So I'll tough it out with this awful user interface that is the web browser.

Leo: Took me years to get you to use Twitter. Now it's going to take me years to get you off it, I know. But that's all right. You don't like change. I got it.

Steve: I still - I do not. I do not like change. My first wife's abbreviation for me was COH, and I was worried to ask her what that stood for.

Leo: What's that stand for?

Steve: Fortunately it stood for Creature of Habit.

Leo: Yes. That's true. That's true.

Steve: So I'm programming in assembly language, darn it. And that's what I learned when I was four, and I'm sticking with it. Anyway, our last break, and then we're going to talk about when heuristics backfire.

Leo: Oh. That's me. Okay. I'm thinking about COH. I kind of am, too, come to think of it, actually. There's nothing wrong with that. You know, you know it well.

Steve: Yes. I only wear - all of our neighbors only see Lorrie and me walking in black.

Leo: Black.

Steve: And they often comment, "Why are you in all black?" It's like, well...

Leo: Easy.

Steve: It's very easy.

Leo: It's always color-coordinated.

Steve: It hides the soy sauce.

Leo: Now, if this were another show, I would have that be the show title, but okay. All right.

Steve: Okay. So Wikipedia defines the term "heuristic." First of all, heuristic comes from the Ancient Greek heurisco, and there it meant "to find or discover." So a heuristic technique is any approach, Wikipedia says, to problem-solving or self-discovery that employs a practical method that is not guaranteed to be optimal, perfect, or rational, but is nevertheless sufficient for reaching an immediate short-term goal or approximation.

Where finding an optimal solution is impossible or impractical, heuristic methods can be used to speed up the process of finding a satisfactory solution. Heuristics can be mental shortcuts that ease the cognitive load of making a decision. Examples that employ heuristics include using trial and error, a rule of thumb, or an educated guess.

Okay. So before we all take some lessons from the implementation of a heuristic that has been causing Microsoft enterprise customers years of mysterious pain, I want to note that I'm a big fan of heuristic methods. I often use heuristics in my own code when I'm dealing with uncertainties. An example from the nearly finished SpinRite 6.1 code comes to mind. Modern operating systems all incorporate the concept of device drivers because the operating system needs to be informed how to talk to the wide range of peripherals it might encounter. For all SpinRites from v1.0 through 6.0, this was never an issue because SpinRite was able to access the system's mass storage hardware through the BIOS which functioned as an abstraction layer. Essentially, it contained permanent built-in device drivers so that it knew how to talk to the hardware on its own motherboard.

The problem was that the BIOS APIs were designed 40 years ago. And mass storage devices have evolved, not only in their capacity, but also in the rich metadata that they're able to offer to anyone who knows how to ask. Unfortunately, the BIOS never did. So to pull off what SpinRite 6.1 needed to do, it could no longer use the BIOS to serve as its intermediary. It needed to talk to the motherboard hardware directly in order to gain direct access to all of the modern hardware capabilities.

But unlike all modern operating systems which are able to use a rich collection of modular device drivers to talk to the hardware, SpinRite needed to have what was essentially a single universal built-in driver that could figure out, for itself and on its own, how to talk to any hardware that it might encounter. And we're talking about 40 years' worth, from long since disappeared manufacturers and so forth. And that's where heuristics comes in.

While SpinRite is starting up, it spends its time literally getting to know the hardware it's running on by performing a series of experiments - in a classic heuristic process - to work out and learn exactly how the underlying hardware operates. It tries things and learns things and keeps a record of what it has learned about every mass storage adapter and device in the system so that it then knows how to interface with each one. There was an appreciable amount of time during these past three years where I wasn't 100% certain that I was going to be able to successfully weave a path through all of the hardware owned by myself and by SpinRite's now 759 individual testers.

But a path was found, and it's been quite a while since any hardware has stumped SpinRite. It doesn't matter how old or new or from which manufacturer. SpinRite 6.1 now has what is essentially a universal smart driver, driven by a heuristic learning process, that's able to interface with any IDE, ATA or AHCI mass storage hardware that has ever been built. And one non-obvious feature of the inherent flexibility of this approach is that the benefit may not only be retrospective. It's likely also prospective, giving SpinRite the ability to figure out things that not only came before it, but may also arrive after it.

So that's an example of the application of heuristics. One way to think of it is as code that figures things out, often working from an initial starting place of uncertainty. The trouble is, being somewhat "rule of thumb" and "inexact," anyone deploying heuristic approaches needs to build in safeguards against their code drawing the wrong conclusions. And this appears to be where a heuristic approach incorporated into Windows Servers by Microsoft back in 2016 - actually it's not just Windows Servers, it's from Windows 10 on, both desktop and server platforms - back in 2016 has fallen down, backfired, and caused unappreciated problems for the past seven years.

Last week, a Norwegian data center engineer named "Simen," spelled S-I-M-E-N, reached out to me via Twitter DM and explained that people who listen to this podcast said that I'd probably be interested in what he had discovered. Given that it became today's topic, it's clear that our podcast listeners have come to know me pretty well. The best way to introduce the seven-year-old problem that Simen uncovered would be to share Ars Technica's coverage of it, which Simen pointed me to in his direct message because it's about him.

So last Wednesday's piece in Ars Technica was titled "Windows feature that resets system clocks based on random data [ouch] is wreaking havoc," and the subtitle was "Windows Secure Time Seeding resets clocks months or years off the correct time." Ars wrote: "A few months ago, an engineer in a data center in Norway encountered some perplexing errors that caused a Windows server to suddenly reset its clock to 55 days in the future. The engineer relied on the server to maintain a routing table that tracked cell phone numbers in real time as they moved from one carrier to the other. A jump of eight weeks had dire consequences because it caused numbers that had yet to be transferred to be listed as having already been moved, and numbers that had already been transferred to be reported as pending.

"The engineer, who asked to be identified only by his first name, Simen, wrote in an email: 'With these updated routing tables, a lot of people were unable to make calls, as we didn't have the correct state. We would route incoming and outgoing calls to the wrong operators. This meant, for example, children could not reach their parents and vice versa.'"

Ars wrote: "Simen had experienced a similar error last August when a machine running Windows Server 2019 reset its clock to January 2023 and then changed it back a short time later."

Leo: Geez.

Steve: I know, Leo. It gets worse. "Troubleshooting the cause of that mysterious reset was hampered because the engineers didn't discover it until after event logs had been purged. The newer jump of 55 days, on a machine running Windows Server 2016, prompted him to once again search for a cause, and this time he found it.

"The culprit was a little-known feature in Windows known as Secure Time Seeding. Microsoft introduced the Secure Time Seeding timekeeping feature in 2016 as a way to ensure that system clocks were accurate. Windows systems with clocks set to the wrong time can cause disastrous errors when they can't properly parse timestamps in digital certificates, or they execute jobs too early, too late, or out of prescribed order. Secure Time Seeding, Microsoft said, was a hedge against failures in battery-powered onboard devices designed to keep accurate time even when the machine is powered down." Right? The old CMOS clock. Remember that? We still have those.

Microsoft engineers wrote - this is Microsoft, or Ars quoting Microsoft: "'You may ask, why doesn't the device ask the nearest time server for the correct time over the network? Since the server is not in a state to communicate securely over the network, it cannot obtain time securely over the network as well, unless you choose to ignore network security, or at least punch some holes into it by making exceptions. To avoid making security exceptions, Secure Time Seeding sets the time based on data inside an SSL handshake the machine makes with remote servers.'"

Okay. So when I read I thought, what? Ars explains some things we know, but adds some new bits. They wrote: "These handshakes occur between two devices connecting

using Secure Sockets Layer protocol, the mechanism that provides encrypted HTTPS sessions. It's also known as Transport Layer Security," they say, as we know. "Because Secure Time Seeding (STS) used SSL certificates in Windows already stored locally, it could ensure that the machine was securely connected to the remote server. 'The mechanism,' Microsoft engineers wrote, 'helped us to break the cyclical dependency between client system time and security keys, including SSL certificates.'"

Okay, now, backing up a little bit, Simen wasn't the only person encountering wild, spontaneous fluctuations in Windows system clocks used in mission-critical environments. Sometime last year, a separate engineer named Ken began seeing similar time drifts. They were limited to two or three servers and occurred every few months. Sometimes, the clock times jumped by a matter of weeks. Other times, the times changed to as late as the year - Leo, this will even be after us - the year 2159.

Leo: That doesn't seem possible.

Steve: I know. It should not be.

Leo: That shouldn't happen.

Steve: So let's just pause here.

Leo: So, oh, it sounds to me - let me see if I understand it. It sounds to me - so you can't - this all makes sense so far, which is if I've lost contact with the outside world, I can't set my clock. But I need to set my clock in order to get outside world access. So I have some internal computers that I'm communicating with over SSL. Yes?

Steve: Okay.

Leo: Does SSL then say, hey, I think it's quarter past four? What do you think? It must be...

Steve: You're going to get to that.

Leo: Okay.

Steve: There is a timestamp in SSL.

Leo: That makes sense. So I don't need, given that I've lost access to the Internet for the time being, I don't need it to be right. I just need to be roughly right. So I'm going to presume that that SSL certificate isn't too far off. So I think this - I understand the logic Microsoft had here. So we're going to get an approximate time, which will then let us fix this problem maybe down the road. But at least we'll be closer than if we're just making it up. Right?

Steve: That's good, yes.

Leo: Okay.

Steve: So unfortunately, and what you described is a useful heuristic with an exception. And we'll see, I mean, like a bad exception. But even so this is an example of a very poorly designed and thought out heuristic. Now, we don't understand yet in detail what's going on. No one does. Microsoft is not telling us, and they don't share their source code. So we don't know exactly what Windows is doing to obtain its time of day and date information. But we don't need to yet. The very fact that it's even possible, as you said, Leo, for a server last year in 2022 to believe that it's now 2159...

Leo: Yeah, that should never happen.

Steve: ...137 years in the future conclusively demonstrates that the designer of this system left out a crucial concept which is very important in heuristic systems. My own name for them is "Sanity Checks," and our listeners will have heard me refer to them from time to time because they're integral to creating robust systems. As the name suggests, a sanity check is a "reasonability filter" that's applied once an answer is obtained from a heuristic system. It prevents believing ridiculous or impossible things.

Okay. So what sanity checks might be applied here? Well, the Microsoft engineers were bragging about how they could rely upon the machine's locally stored TLS certificates to allow them to obtain a secure connection. So would it be reasonable to expect those certificates to have valid expiration dates?

Leo: Yeah.

Steve: Right?

Leo: That'd be reasonable, yeah.

Steve: But 137 years in the future...

Leo: [Crosstalk], couldn't be.

Steve: ...all of those certificates will have expired.

Leo: Yeah.

Steve: So that would appear to fail...

Leo: There's problem number one, yeah. Yeah.

Steve: ...the reasonability test. You know? And what operating Windows server would have all of its certificates expired well more than 100 years ago? Or how about asking the local Windows Update service for the timestamp on the most recently received monthly update?

Leo: There you go.

Steve: Is it reasonable to imagine that the machine has not had a security or feature update in 137 years? Given Microsoft's track record for bugs?

Leo: Maybe. Maybe.

Steve: Uh, no.

Leo: I have a theory in my mind what's happening is you've got two servers that are disconnected. So the first one asks the second one, well, what time is it? And he tells him. And then the second one asks the first one what's the time. And they're advancing each other in a loop till they get hundreds of years in the future.

Steve: What would be bad.

Leo: Your sanity check would have stopped that; right? Just don't keep going. And if it gets too far, you've obviously gone too far. I don't know, they don't - you say they don't know why this is happening.

Steve: We're going to get to it, Leo.

Leo: Okay, then. All right.

Steve: It's worse than you can imagine.

Leo: Oh, dear. Okay.

Steve: So Ken, who also wants to remain anonymous, wrote in an email, but he's been driven crazy by this: "It has exponentially grown to be more and more servers that are affected by this. In total, we have around 20 servers (VMs) that have experienced this, out of 5,000. So it's not a huge amount, but it is considerable, especially considering the damage this does. It usually happens to database servers. When a database server jumps in time, it wreaks havoc," he wrote, "and the backup won't run either, as long as the server has such a huge offset in time. For our customers, this is crucial."

Okay. So gee, seems like the backup system is smart enough to say "Uhhh, something's wrong here." But Windows itself appears to be blissfully ignorant. Simen and Ken, who both asked to be identified only by their first names because they weren't authorized by their employers to speak on the record, soon found that engineers and administrators had

been reporting the same time resets since 2016. In 2017, for instance, a Reddit user in a sysadmin forum reported that some Windows 10 machines the user administered for a university were reporting inaccurate times, in some cases by as many as 31 hours in the past.

Okay. So here again, another missed opportunity for a sanity check. In our current reality, time stubbornly only moves forward. It turns out that this really simplifies things. But that means that discovering that many files in its file system are suddenly timestamped 31 hours in the future might give any well-designed heuristic algorithm pause. In any event, Ars writes: "The Reddit user eventually discovered that the time changes were correlated to a Windows registry key." And then they note HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\SecureTimeLimits.

"Additional investigation showed that the time changes were also linked to errors that reported valid SSL certificates used by the university website were invalid when some people tried to access it." Of course, yeah, because the machine's time was now off.

The admin reached the following conclusion: Windows 10 and 11 have a feature called Secure Time which is on by default. It correlates timestamp metadata from SSL packets, exactly like you've been surmising, Leo, and compares them in time with the machine's local time. Then what this guy wrote is: "It processes these various times by means of 'black magic' and sets the system clock accordingly. Unfortunately, this feature has the tendency to 'flip out' and set the system time to a random time in the past or the future. The flip out might be caused by issues with SSL traffic," this person surmises.

Other examples of people reporting the same behavior date back to 2016, shortly after the rollout of this STS, and many more recent reports of harmful STS-induced time changes have been reported. One Reddit user wrote: "We've run into a show-stopping issue where time on a bunch of production systems jumped forward 17 hours. If you've been in the game more than a week, you know the havoc this can cause."

Okay. So what's going on? To determine the current time, we know two things. STS pulls two pieces of metadata contained in the SSL handshake. It pulls the ServerUnixTime, which is a date and time we've been talking about already this podcast, obtained from the number of seconds that have elapsed since 00:00:00 UTC on January 1st, 1970. The other is cryptographically signed data obtained from the remote server's SSL certificate showing whether it has been revoked under a mechanism we've discussed here at length also, OCSP, the Online Certificate Status Protocol.

So, finally, here's how Ars explains what it was told by Microsoft's engineers. Ars wrote: "Microsoft engineers said they used the ServerUnixTime data 'assuming it is somewhat accurate'" - exactly as you suggested, Leo - "but went on to acknowledge in the same sentence that it 'can also be incorrect.' To prevent STS from resetting system clocks based on data provided by a single out-of-sync remote server, STS makes randomly interspersed SSL connections to multiple servers to arrive at a reliable range for the current time.

"The mechanism then merges the ServerUnixTime with the OCSP validity period to produce the smallest possible time range and assigns it a confidence score. When the confidence score reaches a sufficiently high threshold, Windows classifies the data as STSHC, short for Secure Time Seed High Confidence. The STSHC is then used to monitor system clocks for gross errors and correct them."

So what we've all just heard is a perfect textbook example of a heuristic algorithm. It doesn't get any more heuristic than that. Unfortunately, it is also apparently somehow prone to misfiring badly to become "highly confident" about very wrong times. Ars says: "Despite the checks and balances built into STS to ensure it provides accurate time

estimates, the time jumps indicate the feature sometimes makes wild guesses that are off by days, weeks, months, or even years." Yes, even 137 years.

Ken wrote in his email: "At this point, we're not completely sure why secure time seeding is doing this. Being so seemingly random, it's difficult to understand. Microsoft hasn't really been helpful in trying to track this, either." He said: "I've sent over logs and information, but they haven't really followed up on this. They seem more interested in closing the case."

Simen, meanwhile, said that he's also reported the time resets to multiple groups within Microsoft. When reporting the problems on Microsoft's feedback hub in May, he said he received no company response. He then reported it through the Microsoft Security Response Center in June. The submission was closed as a "non-MSRC case" with no elaboration.

Simen then tapped a third party specializing in Microsoft cloud security to act as an intermediary. The intermediary relayed a response from Microsoft recommending STS be turned off when the server receives reliable timekeeping through the Network Time Protocol. Meaning if NTP, which is what everything has always used, is fine, then use it, and turn off STS, which obviously is able to override that.

Simen wrote in an email: "Unfortunately, this recommendation isn't publicly available, and it is still far from enough to stop the wrongly designed feature to keep wreaking havoc around the world." Simen said he believes the STS design is based on a fundamental misinterpretation of the TLS specification. Microsoft's description of STS acknowledges that some SSL implementations don't put the current system time of the server in the ServerUnixTime field at all. And here it comes, Leo. "Indeed, these implementations, most notably the widely used OpenSSL code library starting in 2014, populate the field with random data."

Leo: Well, that explains it. Oh, boy.

Steve: Wow.

Leo: Well, there you have it.

Steve: So they're getting...

Leo: Geez.

Steve: They're interpreting noise in incoming SSL connections, and every so often enough of the noise happens to line up that they believe a wrong date and jump ahead or forward.

Leo: Because the heuristic will reject a lot of terrible timestamps. It does have some sanity checking. It's just...

Steve: Well, but again, what we're seeing it do is obviously wrong.

Leo: Yeah.

Steve: So it's got some - it has its - basically it's believing its own PR.

Leo: Always a mistake.

Steve: It's receiving nonsense, and every so often the nonsense, you know, the planets align, the nonsense looks like sense, and it goes, oh, my god, it's 137 years earlier than, or, you know, later than I thought. And it jumps the clock forward. That's insane.

Leo: Yeah. Well, they misinterpreted the TLS spec, that's all. Or made assumptions about it that were incorrect.

Steve: And then failed to put a reasonability filter on the result of their heuristic.

Leo: Well, but also, well, I guess, I mean, they maybe don't want to admit it because it might break the whole STS mechanism. If you can't trust what you're getting back, this whole system might be a faulty system.

Steve: The whole system is a faulty system.

Leo: Yeah.

Steve: Basically, this is someone's bad design from...

Leo: Well, it seemed like a good idea.

Steve: That they're in love with from seven years ago.

Leo: Right. They can't let go of it.

Steve: And it is just effing things up, and it's causing real damage to their customers. And Microsoft doesn't want to hear about it. They like it. You know, I mean, I'm sure it got elevated to someone who said, no, no, it works. It's got heuristics, you know.

Leo: Well, what would be worse, not having any system for correcting the time, which means you never have any hope of getting a time from an NTP server maybe, or trying to get something? They just need better heuristics; right? That's really the...

Steve: Yes, this is bad.

Leo: It's not bad in theory, but knowing now that they could get random data, they need a much better...

Steve: And Leo, they could have fixed it seven years ago.

Leo: Yeah. Well, I'm trying to think that maybe they thought, well, we can't abandon this. It's better than nothing. But obviously it's not better than nothing. So, yeah.

Steve: No. We had working systems before 2016.

Leo: Right.

Steve: Before they did this. Things were fine. They weren't wreaking this kind of havoc.

Leo: They need a better heuristic.

Steve: So Simen said: "The false assumption is that most SSL implementations return the server time."

Leo: Right.

Steve: "This was probably true in a Microsoft-only ecosystem back when they implemented it."

Leo: Because there was no OpenSSL being run, yes, right.

Steve: "But at that time, when STS was introduced, OpenSSL was already sending some random data." Okay. So now...

Leo: Maybe they're hoping to get the OpenSSL guys to fix their code. Maybe they think it's their problem.

Steve: It's probably a privacy problem. I mean, I imagine OpenSSL is randomizing this because first of all it doesn't, obviously it doesn't have to be right because OpenSSL works just fine.

Leo: Right.

Steve: And it's probably leaking information about the exact time that the OpenSSL server thinks it is, and you could use that in order to, like, to do some deanonymization stuff.

So while official Microsoft talking points play down the unreliability of STS, Ryan Ries, whose LinkedIn profile indicates he is a senior Windows escalation engineer at Microsoft, wasn't as reticent when discussing STS on Twitter last year. And I have in the show notes a copy of Ryan's posting on Twitter. He says, and he's got a big beard, so you know you can believe him...

Leo: It's got to be true, yeah.

Steve: Big gray beard.

Leo: Yeah, yeah.

Steve: He says: "Hey, people. If you manage Active Directory domain controllers, I want to give you some UNOFFICIAL [that's in all caps] advice that is solely my person opinion: Disable Secure Time Seeding for w32time on your domain controllers."

Leo: Ha-hah. Well, there you go.

Steve: There it is. January 31st, 2022. So that was - he was heading in to New Year's Eve, and he thought, you know, let's just give everybody a head's up for the New Year.

So someone named Brian Clark replied to Ryan's tweet. He said: "And why do you think we should do this?" And Ryan says: "Because it's just a matter of time - wink - before it bites you in the butt."

Leo: Wow. So they've known about this.

Steve: Yup.

Leo: Made no attempt to fix it.

Steve: Nope.

Leo: [Heavy sigh]

Steve: So again, this was a senior Windows escalation engineer at Microsoft.

Leo: He'd clearly see it happen. He'd seen it happen. He knew.

Steve: Yup. Of course. Several hours after Ars - I think this is what's happened to Microsoft, Leo. It has become too large and sprawling. It's just no longer responsible for itself. Anyway, several hours after Ars first posted this, a Microsoft representative emailed Ars the following statement. This is from Microsoft: "Secure Time Seeding

feature is a heuristic-based method of timekeeping that also helps correct system time in case of certain software/firmware/hardware timekeeping failures. The feature has been enabled by default in all default Windows configurations and has been shown to function as intended in default configurations.

"Time distribution is unique to each deployment, and customers often configure their machines to their particular needs. Given the heuristic nature of Secure Time Seeding and the variety of possible deployments used by our customers, we have provided the ability to disable this feature if it does not suit your needs. Our understanding is that there are likely unique, proprietary, complex factors in deployments where customers are experiencing Secure Time Seeding issues, and these customers do not benefit from this feature" - right - "as it is currently implemented." Of course they're not going to change the implementation apparently. "In these isolated cases, the only course of action we can recommend is to disable this feature in their deployments." In other words...

Leo: It's your fault for using OpenSSL is what they're saying, I think.

Steve: Yup. For having it around.

Leo: Yup.

Steve: Exactly. And they said: "We agree that the overall direction of technology with the adoption of TLS v1.3 and other deployments in this area could make Secure Time Seeding decreasingly effective over time." What do you want to bet, Leo, that TLS 1.3 explicitly randomizes that field for the sake of privacy, and this is the way they're saying it. "We agree that the overall direction of technology" - Jesus - "with the adoption of TLS 1.3 and other developments in this area could make Secure Time Seeding decreasingly effective over time."

Leo: But that's not our fault.

Steve: That's right. "But we are not aware of any bugs arising from their use."

Leo: It would be possible to build in a sanity check that would at least prevent the most egregious...

Steve: Yes. Yes.

Leo: So they could fix this.

Steve: Leo, look at the recent Windows Update time. When did the most recent Windows Updates occur? Was it 137 years ago that you updated Windows?

Leo: Yeah. There's ways to look at this.

Steve: Unlikely. Was it six months? No. It was, you know, within a month or two. And you've got - you have file system activity.

Leo: Yeah.

Steve: You know? Is it reasonable that no files have been written in the last month on a domain controller?

Leo: Well, can you query - there are things you could query about that; right?

Steve: Yes.

Leo: Because we can presume that the machine I'm using is not accurate.

Steve: Time is littered, time is, you know, machines are littered with time clues.

Leo: Interesting. Seems like this is solvable. But they may have said, you know, it's just going to get worse and worse. And increasingly, especially with TLS 1.3, we're not going to get any real-time signals that we can trust. So we just want people to turn it off.

Steve: So we're going to keep using them.

Leo: Yeah, well, that's not the right answer, obviously.

Steve: So the well-known personality HD Moore, you know, the CTO and co-founder at runZero who also developed the Metasploit Project Foundation, the pen-testing suite, he speculated on Signal that the cause is some sort of logic bug in Microsoft's code.

Leo: Yeah.

Steve: So HD Moore wrote: "If OpenSSL has been setting random Unix times in TLS responses for a long period of time, but this bug is showing up infrequently, then it's likely harder to trigger than just forcing a bunch of outbound TLS connections to a server with bogus timestamp replies. If it was that easy, it would happen far more frequently. Either the STS logic relies on different root certificates as the signer, or some variety in the hostnames/IPs, or only triggers on certain flavors of random timestamp like values dividable by 1024 or something. It smells like a logic bug that is triggered infrequently by fully random timestamps and likely just some subset of values and with some other conditions, like multiple requests in some period of time to multiple certs, et cetera."

And then Ars wraps up their coverage of this by writing: "As the creator and lead developer of the Metasploit exploit framework, a pen tester, and chief security officer, Moore has a deep background in security. He speculated that it might be possible for malicious actors to exploit STS to breach Windows systems that don't have STS turned

off. One possible exploit would work with an attack technique known as Server Side Request Forgery.

"Microsoft's repeated refusal to engage with customers experiencing these problems means that, for the foreseeable future, Windows will by default continue to automatically reset system clocks based on values that remote third parties include in SSL handshakes. Further, it means that it will be incumbent on individual admins to manually turn off STS when it causes problems." Of course by that time it's too late because it caused a problem, which could be really bad.

Leo: Yeah.

Steve: "That in turn," they said, "is likely to keep fueling criticism that the feature as it has existed for the past seven years does more harm than good."

Leo: Wow.

Steve: Simen wrote: "STS is more like malware than an actual feature."

Leo: Holy cow.

Steve: He said: "I'm amazed that the developers didn't see it, that QA didn't see it, and that they even wrote about it publicly without anyone raising a red flag. And that nobody at Microsoft has acted when being made aware of it."

Leo: Yeah.

Steve: So, wow. In the registry, HKEY_LOCAL_MACHINE\SYSTEM\

CurrentControlSet\Services\W32Time\Config contains a REG_DWORD value named UtilizeSslTimeData. That makes it pretty clear. And now that we know that the SSL time data that's present in OpenSSL's handshakes is random noise, that doesn't seem like a good thing to leave enabled. In my Win10 machine, that value was set to one to enable this feature in Windows. It is now set to zero, and anyone else who's concerned by this can set theirs to zero. After rebooting Windows, it will rely upon your local machine's clock and the reliable Network Time Protocol to set it and keep it set correctly. End of story.

Leo: Well, well, well.

Steve: I started out observing heuristic rules of thumb approaches can be extremely valuable. They make SpinRite 6.1 possible as it is. But they need to be designed with care and protected from going badly wrong. It appears that Microsoft has done neither of these things in a system that has already and may still damage their users.

Leo: Jiminy Christmas.

Steve: And they don't care.

Leo: It's a fascinating case study, though, in attempting to solve a problem with a very kind of clever hack, but not - kind of not doing it right, I guess.

Steve: Yeah.

Leo: This is the answer. And that's the problem with clever hacks.

Steve: I would have titled this podcast "You're doing it wrong."

Leo: You're doing it wrong. Wow. And seemed to be unwilling to fix it, which is even weirder. Wow. Very good...

Steve: You know?

Leo: Really good case study, I think; you know? I like this idea that you should start doing like a professor of law, you know, where you come up with a case study, and what do we learn from this. I love this. You could do more of that. It's fun, it's really fun.

Steve: I know that our listeners do, too.

Leo: Yeah, really good stuff.

Steve: So when I can, I certainly will.

Leo: Yeah, thank you, Steve. Steve Gibson is at GRC.com. That's his website, the Gibson Research Corporation. Well, you'll find many wonderful things there. Of course SpinRite is his bread and butter, the world's best mass storage maintenance and recovery utility. Perfect for your failing SanDisk SSD.

Steve: Bye.